Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 20 (2013) 39 – 45

Procedia Computer Science

Complex Adaptive Systems, Publication 3
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2013- Baltimore, MD

Performance Analysis of Kernel Adaptive Filters Based on LMS Algorithm

Ibtissam Constantin[a,] *, Régis Lengellé[b]

[a]*Lebanese University, Faculty of Science, Fanar, Lebanon*
[b]*Laboratory ISTIT -LM2S(FRE CNRS 2732) Troyes University of Technology, Troyes, France*

Abstract

The design of adaptive nonlinear filters has sparked a great interest in the machine learning community. The present paper aims to present some recent developments in nonlinear adaptive filtering. It provides an in-depth analysis of the performance and complexity of a class of kernel filters based on the least-mean-squares algorithm. A key feature that underlies kernel algorithms is that they map the data in a high-dimensional feature space where linear filtering is performed. The arithmetic operations are carried out in the initial space via evaluation of inner products between pairs of input patterns called kernels. The SNR improvement and the convergence speed of kernel-based least-mean-squares filters are evaluated on two types of applications: time series prediction and cardiac artifacts extraction from magnetoencephalographic data.

Selection and peer-review under responsibility of Missouri University of Science and Technology

Keywords: Adaptive kernel filters; LMS algorithm; SNR improvement; cardiac artifacts extraction; time series prediction

1. Introduction

Linear filters have played a crucial role in the development of various signal processing techniques. The obvious advantage of linear filters is their inherent simplicity. Design and analysis of such filters are relatively straightforward tasks in many applications. However, there are many practical situations that require nonlinear processing of the signals involved. A great deal of research effort was expended on the development of nonlinear filters. Unlike linear filters which are characterized by their impulse response, there exists a whole variety of system representations for nonlinear filters. Two models have prevailed in the literature. Polynomial filters, usually called Volterra series based filters [1] and neural networks [2]. Volterra filters can model a large class of nonlinear systems. They are attractive because their output is expressed as a weighted linear sum of monomial functions of the input patterns which simplifies the filter design. However, their primary disadvantage is their computational complexity that explodes exponentially as their modeling capacity increases. Neural networks provide powerful tools for solving general nonlinear filtering problems. They are endowed with the universal approximation property [2] in the sense that they can approximate any continuous multivariate function to any degree of accuracy, provided that sufficiently many hidden neurons are available. However, they are hindered by their nonconvex optimization nature and the risk of being stuck in local minima.

Kernel algorithms, based on the theory of reproducing kernel Hilbert spaces (RKHS), represent an emerging technology rooted in many fields such as classification, regression and pattern recognition. In the context of filtering, an optimum Wiener filter based on kernels has been developed in previous work [3]. Adaptive kernel filters, based on the recursive least-squares (RLS) and least-mean-squares (LMS) algorithms have also been designed [4]-[11]. Optimum and adaptive filters have shown outstanding performances in many applications. The key idea that underlies kernel algorithms is to map the data in a high-dimensional feature space where linear algorithms are applied. The arithmetic operations are carried out in the initial space via evaluation of inner products between pairs of input patterns called kernels. The main problem that arises in kernel methods is their linear growing structure with each new sample, which enforces the need for sparsification techniques. Sparsification aims at discarding training samples that do not affect model performance. Several sparsification techniques have been developed to

* Corresponding author. Tel.: (961) 70 58 52 60; fax: (961)1686994.
E-mail address: ibtissamconstantin@ul.edu.lb.

Selection and peer-review under responsibility of Missouri University of Science and Technology
doi:10.1016/j.procs.2013.09.236

curtail the growth of kernel models including the approximate linear dependency (ALD) [4], the coherence criterion [10], the novelty criterion [11] and the surprise criterion [11]. A kernel adaptive filter embodies a two-step process: a sparsification step that removes undesired training data and a parameter update step.

The present paper pertains to the study of kernel-based adaptive LMS filtering. LMS filters have the advantage of having lower computational complexity per update and being simpler to implement than RLS filters. One may identify two types of kernel LMS filters that differ in the sparsification step: the LMS filter developed by Liu et al. designated by KLMS [11] discards totally the redundant training data and does not use it to adjust the filter parameters whereas in SSP (sparse sequential projection) [6,7] and NORMA filters [8,9] the redundant training data is incorporated in the parameter update mechanism. This study investigates algorithm implementation, performance and complexity issues related to these filters. It is organized as follows. Section 2 introduces the theoretical concepts of optimal filtering in RKHS. Section 3 provides a brief review of the existing sparsification schemes developed to control the model growth. Section 4 describes the kernel-based LMS algorithms in fair details and analyzes their complexity. Finally, experiments involving both synthetic and real data are reported in section 5. Some concluding remarks are given in section 6.

2. Principles of kernel-based optimal filtering in RKHS

A possible way to design nonlinear filters is to map the input data $\boldsymbol{x}_i$ into a high-dimensional space using a nonlinear function $\boldsymbol{\varphi}(.)$ and apply linear filtering techniques. However, the dual space may have very high, even infinite dimension which renders this approach unfeasible in practice. Kernel methods achieve an implicit mapping of the data by using the theory of reproducing kernel Hilbert spaces. The key idea is that a Mercer kernel function represents a dot product in a RKHS. Thus any algorithm that can be expressed in terms of dot products can be extended to nonlinear processing by replacing each dot product by a Mercer Kernel. The next section briefly reviews the main definitions and properties related to reproducing kernel Hilbert spaces and Mercel kernels [12].

2.1. RKHS and Mercer kernels

Let $\mathcal{H}$ be a reproducing kernel Hilbert space of real-valued functions $\psi(.)$ on a compact $\chi \subset \mathcal{R}^l$, and let $<.,.>_{\mathcal{H}}$ be the inner product in $\mathcal{H}$. By virtue of the Riesz representation theorem, there exists a unique function $\kappa(.,\boldsymbol{y})$ of $\mathcal{H}$, called *representer of evaluation* at $\boldsymbol{y}$, which verifies the following reproducing property [12]

$$\psi(\boldsymbol{y}) =< \psi, \kappa(.,\boldsymbol{y}) >_{\mathcal{H}},\ \forall\ \psi \in \mathcal{H} \tag{1}$$

for every $\boldsymbol{y} \in \chi$. Replacing $\psi(.)$ by $\kappa(.,\boldsymbol{x})$ in (1) yields

$$\kappa(\boldsymbol{x},\boldsymbol{y}) =< k(.,\boldsymbol{x}), \kappa(.,\boldsymbol{y}) >_{\mathcal{H}} \tag{2}$$

for every $\boldsymbol{x},\boldsymbol{y} \in \chi$. Equation (2) is the origin of the now generic term reproducing kernel to refer to $\kappa(.,.)$. Note that $\mathcal{H}$ can be restricted to the span of $\{\kappa(.,\boldsymbol{y}), \boldsymbol{y} \in \chi\}$ because according to (1) nothing outside this set affects $\psi(.)$ evaluated at any point of χ. Denoting by $\varphi(.)$ the map that assigns to each input $\boldsymbol{y}$ the kernel function $\kappa(.,\boldsymbol{y})$, (2) implies $\kappa(\boldsymbol{x},\boldsymbol{y}) =< \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) >_{\mathcal{H}}$. The kernel then evaluates the inner product of any pair of elements of χ mapped to $\mathcal{H}$ without any explicit knowledge of either $\varphi(.)$ or $\mathcal{H}$. This idea is known as the *kernel trick*. Of course, not every function can be considered a kernel. According to Mercer theorem [12] the following condition should be fulfilled

$$\iint \kappa(\boldsymbol{x},\boldsymbol{y}) g(\boldsymbol{x}) g(\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \geq 0 \tag{3}$$

for all g satisfying $\int g(x)^2\, dx < \infty$. Classical examples of kernels are the radial Gaussian kernel $\kappa(\boldsymbol{x},\boldsymbol{y}) = exp(-\| \boldsymbol{x}-\boldsymbol{y} \|^2/2\beta^2)$, with β >0 and the polynomial kernel $\kappa(\boldsymbol{x},\boldsymbol{y}) = (\alpha+< \boldsymbol{x},\boldsymbol{y} >)^q$, with $\alpha \geq 0$ and $q \in \mathbb{N}^*$.

2.2. Optimal filtering in RKHS

Let $\mathcal{H}$ be a RKHS defined by a kernel κ . The least-squares problem in $\mathcal{H}$ consists in finding a function $\psi(.)$ minimizing the sum of the squared errors between n samples d_i of the desired response and the corresponding model output samples $\psi(\boldsymbol{x}_i) =< \psi(.), \kappa(.,\boldsymbol{x}_i) >_{\mathcal{H}}$, namely

$$\min_{\psi \in \mathcal{H}} \sum_{i=1}^{n} (d_i - \psi(\boldsymbol{x}_i))^2. \tag{4}$$

By virtue of the representer theorem [13], the function $\psi(.)$ of $\mathcal{H}$ minimizing (4) can be written as a kernel expansion in terms of available data

$$\psi(.) = \sum_{j=1}^{n} \alpha_j \kappa(.,\boldsymbol{x}_j). \tag{5}$$

It can be shown that (5) becomes $\min_{\alpha} \| \boldsymbol{d} - \boldsymbol{K\alpha} \|^2$, where **K** is the Gram matrix whose (i,j)th entry is $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The solution vector $\boldsymbol{\alpha}$ is found by solving the *n*-by-*n* linear system of equations $\boldsymbol{K\alpha} = \boldsymbol{d}$. The main concern that arises from this result is how to process an increasing amount of observations and update the model (5) in online applications. Before proceeding to the analysis of LMS kernel-based techniques, which are the topic of this study, a review of the various sparsification techniques developed in the literature is outlined.

3. A brief review of sparsification rules

The aim of sparsification is to produce a reduced order model of the form $\psi_n(.) = \sum_{j=1}^{m} \alpha_{n,j} \kappa(., \boldsymbol{c}_j)$, where the set $\mathcal{C}_n = \{\boldsymbol{c}_j\}_{j=1}^{m}$ is referred to as *dictionary*. The centers $\boldsymbol{c}_j$ are picked from the input samples. In [8,9], an update model parameters procedure was designed, in which the old parameters $\alpha_{n,j}$ vanish over time by effect of regularization. Samples associated with zero $\alpha_{n,j}$ were removed from the dictionary. Another approach that exploits the concept of approximate linear dependency (ALD) was considered in [4]. If a kernel function $\kappa(., \boldsymbol{x}_n)$ at time *n* can be approximately represented by a linear combination of previously selected elements, $\boldsymbol{x}_n$ is not inserted in the dictionary. ALD tests if the following condition is satisfied

$$\min_{\boldsymbol{a}} \| \kappa(., \boldsymbol{x}_n) - \sum_{\boldsymbol{c}_j \in \mathcal{C}_{n-1}} a_j \, \kappa(., \boldsymbol{c}_j) \|^2 \leq \delta, \tag{6}$$

where δ is a preset threshold that determines the level of sparsity of the model. A similar criterion was considered in [6,7] but in different form. It computes the difference between the updated model and its orthogonal projection onto the existing function space. If this difference is smaller than δ, the sample $\boldsymbol{x}_n$ is discarded. Richard *et al.* [10] proposed the coherence parameter that measures, like ALD, the dependence of the elements in the dictionary with lower computational cost. In [11], a simple yet effective criterion, the novelty criterion, that computes the distance measure in the input space instead of the feature space, was investigated. The novelty criterion, originally introduced by Platt [14], tests the following condition

$$\min_{\boldsymbol{c}_j \in \mathcal{C}_{n-1}} \| \boldsymbol{x}_n - \boldsymbol{c}_j \|^2 \leq \delta. \tag{7}$$

An information theoretic criterion called surprise was also devised in [11] based on Gaussian processes theory. The ALD, coherence and novelty criterion can be viewed as special cases of the surprise criterion [11].

4. Kernel-based adaptive LMS filtering

A kernel-based algorithm involves two procedures: Sparsification and filter parameters update. LMS kernel-based algorithms adapts the filter parameters using a stochastic gradient approximation in RKHS. There exists two types: KLMS [11] removes totally the redundant training data whereas SSP [6,7] and NORMA [8,9] utilizes the redundant data to update the filter parameters. The next section studies the performance and complexity of these algorithms.

4.1. KLMS

KLMS [11] minimizes the empirical risk (4) using stochastic gradient descent. It performs gradient descent with respect to the instantaneous squared error $(d_n - \psi(\boldsymbol{x}_n))^2$. The general form of the update rule is:

$$\psi_n = \psi_{n-1} - \eta_n \partial_{\psi} (d_n - \psi(\boldsymbol{x}_n))^2 \big|_{\psi=\psi_{n-1}}, \tag{8}$$

where ∂_{ψ} is shorthand for $\frac{\partial}{\partial_{\psi}}$ and η_n is the learning rate. By virtue of property (1), $\partial_{\psi} (d_n - \psi(\boldsymbol{x}_n))^2 \big|_{\psi=\psi_{n-1}} = -2(d_n - \psi(\boldsymbol{x}_n)\kappa(., \boldsymbol{x}_n)$. Therefore, the previous equation can be written as

$$\psi_n = \psi_{n-1} + \eta_n e_n \, \kappa(., \boldsymbol{x}_n) \, , \tag{9}$$

where $e_n = d_n - \psi(\boldsymbol{x}_n)$. With the arrival of new samples, the kernel expansion (9) will increase linearly. In [11], the novelty and surprise criteria were used to determine whether each incoming new sample should be included or discarded. KLMS algorithm removes redundant samples without updating the parameters of the kernel expansion.

4.2. SSP

SSP [6,7] adopts the same update rule (9). However sparsification is accomplished in a different manner. Suppose that the model at time *n-1* has the form:

$$\psi_{n-1}(.) = \sum_{j=1}^{m-1} \alpha_{n,j} \kappa(., \boldsymbol{c}_j), \tag{10}$$

where the centers $\boldsymbol{c}_j$ have been selected among the input samples. Upon the arrival of a new sample $\boldsymbol{x}_n$ at time *n*, SSP computes the orthogonal projection of ψ_n onto the subspace $\mathcal{H}_{m-1}$ spanned by the kernel functions $\kappa(., \boldsymbol{c}_j), j = 1, \dots, m-1$. If $\| \psi_n - \psi_n^{\perp} \|^2 > \delta$, where $\psi_n^{\perp}$ is the projection of ψ_n and δ is a preset threshold, $\boldsymbol{x}_n$ is admitted in the dictionary and the model is updated with ψ_n . Otherwise the input $\boldsymbol{x}_n$ is discarded and the model is updated with $\psi_n^{\perp}$. Although $\boldsymbol{x}_n$ is considered as redundant for the dictionary, it may bear useful information for adjusting the filter parameters.

The main problem is centered on how to compute $\psi_n^{\perp}$. Since $\psi_n - \psi_n^{\perp} \perp \mathcal{H}_{m-1}$, the following equality is deduced

$$\langle \kappa(., \boldsymbol{c}_j), \psi_n - \psi_n^{\perp} \rangle_{\mathcal{H}_{m-1}} = 0, \; j = 1, \dots, m-1. \tag{11}$$

Moreover $\psi_n^{\perp} \in \mathcal{H}_{m-1}$. It can be expressed as

$$\psi_n^{\perp}(.) = \sum_{j=1}^{m-1} \beta_{n,j} \kappa(., \boldsymbol{c}_j), \tag{12}$$

where $\beta_{n,j}$ are the unknown parameters. Combining (9) and (10) yields

$$\psi_n(.) = \sum_{j=1}^{m} \alpha_{n,j} \kappa(., \boldsymbol{c}_j), \tag{13}$$

where $\kappa(., \boldsymbol{c}_m) = \kappa(., \boldsymbol{x}_n)$ and $\alpha_{n,m} = \eta_n e_n$. Substituting equations (12) and (13) into (11) and using (2) leads to

$$\boldsymbol{K}_{n-1} \boldsymbol{\beta}_n = \boldsymbol{\gamma}\_n \,, \tag{14}$$

where $\boldsymbol{K}_{n-1}$ is the (*m-1*)- by-(*m-1*) Gram matrix whose (*i,j*)th entry is $\kappa(\boldsymbol{c}_i, \boldsymbol{c}_j)$, $\boldsymbol{\gamma}_n = [\boldsymbol{K}_{n-1} \quad \boldsymbol{h}_n] \boldsymbol{\alpha}_n$ and $\boldsymbol{h}_n = [\kappa(\boldsymbol{c}_1, \boldsymbol{x}_n) \quad \kappa(\boldsymbol{c}_2, \boldsymbol{x}_n) \quad \dots \quad \kappa(\boldsymbol{c}_{m-1}, \boldsymbol{x}_n)]^t$.

Using (12) and (13), it can be shown that :

$$\| \psi_n - \psi_n^{\perp} \|^2 = \boldsymbol{\alpha}_n^t \begin{bmatrix} \boldsymbol{K}_{n-1} & \boldsymbol{h}_n \\ \boldsymbol{h}_n^t & \kappa(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix} \boldsymbol{\alpha}_n - \boldsymbol{\alpha}_n^t \begin{bmatrix} \boldsymbol{K}_{n-1} \\ \boldsymbol{h}_n^t \end{bmatrix} \boldsymbol{\beta}_n. \tag{15}$$

The computation of $\boldsymbol{\beta}_n$ requires the inversion of the Gram matrix. Note that $\boldsymbol{K}_n = \begin{bmatrix} \boldsymbol{K}_{n-1} & \boldsymbol{h}_n \\ \boldsymbol{h}_n^t & \kappa(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}$. Its inverse can be computed recursively using the block matrix inversion lemma [11]:

$$\boldsymbol{K}_n^{-1} = \boldsymbol{\sigma}_n^{-1} \begin{bmatrix} \boldsymbol{K}_{n-1}^{-1} \boldsymbol{\sigma}_n + \boldsymbol{u}_n \boldsymbol{u}_n^t & -\boldsymbol{u}_n \\ -\boldsymbol{u}_n^t & 1 \end{bmatrix}, \tag{16}$$

where $\boldsymbol{u}_n = \boldsymbol{K}_{n-1}^{-1} \boldsymbol{h}_n$ and $\boldsymbol{\sigma}_n = \boldsymbol{\kappa}(\boldsymbol{x}_n, \boldsymbol{x}_n) - \boldsymbol{u}_n^t \boldsymbol{h}_n$.

4.3. NORMA

NORMA [8,9] minimizes the following empirical risk

$$\min_{\psi \in \mathcal{H}} \sum_{i=1}^{n} (d_i - \psi(\boldsymbol{x}_i))^2 + \lambda \| \psi \|^2, \tag{17}$$

where a regularization parameter has been added in order to prevent the algorithm from overfitting the data. As KLMS and SSP, NORMA performs stochastic gradient descent with respect to the instantaneous risk $(d_n - \psi(\boldsymbol{x}_n))^2 + \lambda \parallel \psi \parallel^2$. The gradient is defined as $\partial_\psi (d_n - \psi(\boldsymbol{x}_n))^2 + \lambda \parallel \psi \parallel^2 \big|_{\psi=\psi_{n-1}} = -2(d_n - \psi(\boldsymbol{x}_n)\kappa(.,\boldsymbol{x}_n) + 2\lambda\psi$, which leads to the following update rule

$$\psi_n = (1 - \eta_n\lambda)\psi_{n-1} + \eta_n e_n \, \kappa(.,\boldsymbol{x}_n). \tag{18}$$

At each iteration, a kernel function is inserted and the parameters of the previous model are truncated by $(1 - \eta_n\lambda)$. Sparsity is achieved by dropping small coefficients.

4.4. Computational complexity

Among the three algorithms, SSP has the highest computation cost. The sparsification procedure in SSP requires the inversion of the kernel matrix which is performed with $O(m^2)$ operations. The complexity of the parameter update is $O(m)$. This leads to an overall complexity of $O(m^2)$. The complexity of KLMS is clearly lower than SSP if an appropriate sparsification criterion to build the dictionary is chosen. For instance, the novelty criterion may be selected which requires $O(m)$ arithmetic operations. The adjustment of the model parameters, which is only accomplished when a new sample is inserted in the dictionary has $O(m)$ complexity. However, as it will be seen in the experimental section, the size of the kernel expansion in SSP is significantly lower than KLMS. It never exceeds a few tens. NORMA is an $O(m)$ algorithm. The experimental section shows that the use of even small values of the regularization parameter degrades the performance of the algorithm when compared to KLMS and SSP. The size of the kernel expansion in NORMA is equal to the size of the training data.

5. Experiments

Experiments were conducted on simulated and real data to compare the performance of the proposed techniques. The Gaussian kernel was considered in all the experiments. The enhanced novelty criterion was used for KLMS algorithm [11]. It tests first if the condition (7) is verified. If it is, the input sample is discarded. Otherwise it tests if the prediction error exceeds a threshold μ. If this condition is met, the input sample is accepted as a new center.

The first application consists of predicting the highly nonlinear time series [15]

$$d_n = (0.8 - 0.5\exp(-d_{n-1}^2))\, d_{n-1} - (0.3 + 0.9\exp(-d_{n-1}^2))d_{n-2} + 0.1\sin(\pi d_{n-1})). \tag{19}$$

The data were generated by iterating the above equation from the initial conditions (0.1,0.1). Output d_n was corrupted by an additive zero-mean white Gaussian noise with variance σ^2. Different realizations of the noise were considered as depicted in table 1. The time embedding were chosen as 2, i.e, $\boldsymbol{x}_n = [x_{n-2} \quad x_{n-1}]^t$ was used as the input to predict $\boldsymbol{x}_n$. A variable step size parameter was used that is updated according to [16]

$$\eta_n = \alpha\eta_{n-1} + \gamma e_{n-1}^2, \tag{20}$$

where $0 < \alpha < 1$ and $\gamma > 0$, and η_n is set to η_{min} and η_{max} when it falls below or above these bounds. This step size induced better performance than a fixed step size. As can be seen from (20), a large prediction error increases the step size to provide faster tracking. If the prediction error decreases, the step size will be decreased to reduce the misadjustment. Preliminary experiments yielded $\alpha = 0.99$, $\gamma = 0.001$, $\eta_{min} = 0.01$, and $\eta_{max} = 1$. A test set of 100 sequences of 4000 samples was constituted and the performance of the algorithms was measured in steady state by evaluating the normalized mean-square error (NMSE) over the last 1000 samples of each sequence, namely

$$\frac{1}{1000\sigma_d^2}\sum_{n=3001}^{4000}(d_n - \psi_{n-1}(\boldsymbol{x}_n))^2, \tag{21}$$

where σ_d is the variance of the desired output in the selected interval. The optimum parameters setting was determined by cross-validation. The parameters δ and μ were chosen by grid search over $(10^{-4} \le \delta \le 10^{-1}) \times (10^{-4} \le \mu \le 10^{-1})$ with increment 2×10^{-k} within each range $[10^{-k} \quad 10^{-k+1}]$. The kernel parameter β was varied from 0.1 to 0.9 in increments of 0.05. 25 sequences of 4000 samples were generated and the average NMSE was computed on the last 1000 samples for each combination of (δ, μ, β). It was observed that the regularization parameter λ deteriorates the performance of NORMA compared to SSP and KLMS and therefore was set to 0. Table 1 reports the average NMSE and the average dictionary size on the test set for different noise variances σ^2. It shows that the three methods exhibit comparable performances, with a slight advantage for KLMS. However, in terms of sparsity, NORMA and KLMS require significantly higher number of centers than SSP. This could be expected since the regularization parameter in Norma is equal to 0 and that KLMS removes totally the redundant training data. KLMS included most of the training samples in the dictionary because they are needed in the parameter update. Figure 1

displays the learning curves for $\sigma^2 = 0.01$. The three methods have similar convergence rates, requiring about 500 iterations to converge.

Table 1. NMSE and dictionary size for different realizations of the noise, averaged on 100 sequences.

	KLMS		SSP		NORMA	
σ	NMSE	Dictionary size	NMSE	Dictionary size	NMSE	Dictionary size
0.5	0.158	1280.8	0.160	72.99	0.160	4000
0.1	0.035	1826	0.036	33.78	0.036	4000
0.06	0.0136	976.12	0.0138	31.19	0.0138	4000

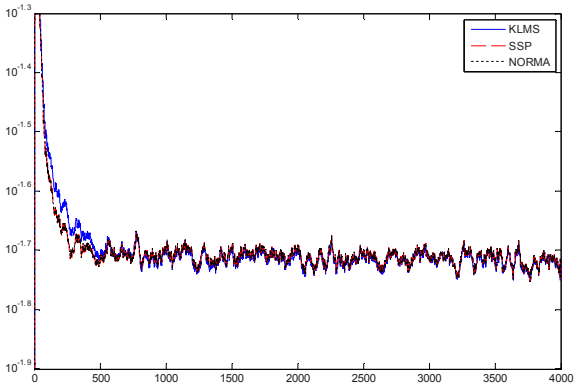

Fig.1. Learning curves for KLMS, SSP and NORMA obtained by averaging over 100 experiments.

As a second application, KLMS, SSP and NORMA were applied on magnetoencaphalographic (MEG) data corrupted by cardiac artifacts. The recorded electocardiographic (ECG) signal was used as the reference signal. The MEG signal was used as the desired output so the residue is the denoised MEG signal. NMSE was used as a measure of performance. A variable step size parameter was selected that obeys the same update rule (20). After trial and error, the values $\alpha = 0.998$, $\gamma = 0.00005$, $\eta_{min} = 0.0001$, and $\eta_{max} = 0.9$ were selected. The optimum parameters δ, μ and β were determined by cross-validation as previously by performing a grid search over $(10^{-4} \le \delta \le 10^{-1}) \times (10^{-4} \le \mu \le 10^{-1}) \times (10^{-2} \le \beta \le 1)$ with increment $2 \times 10^{-k-1}$ within each range $[10^{-k}\ \ 10^{-k+1}]$ for δ and μ and $5\times 10^{-k-1}$ for β. The dimension of the input vector l was varied from 2 to 13. Here again, the regularization parameter λ was set to 0. KLMS, SSP and NORMA were tested on 10 independent sequences of 10000 samples and NMSE was computed on the last 4000 samples of each sequences. Table 2 summarizes the results, averaged over the 10 independent trials. We notice that SSP produced a much sparser model than KLMS and NORMA with equivalent performance. Figure 2 presents SSP results for one trial, where l=11, $\beta = 0.15$ and $\delta = 0.0003$. The highest two curves show the contaminated and denoised MEG signals. Curves C1 and C2 show the ECG reference channel and the estimated artifact signal.

Table 2. NMSE and dictionary size obtained with MEG data averaged on 10 sequences.

Algorithm	NMSE	Dictionary size
KLMS	0.853	9186.2
SSP	0.854	16.8
NORMA	0.856	10000

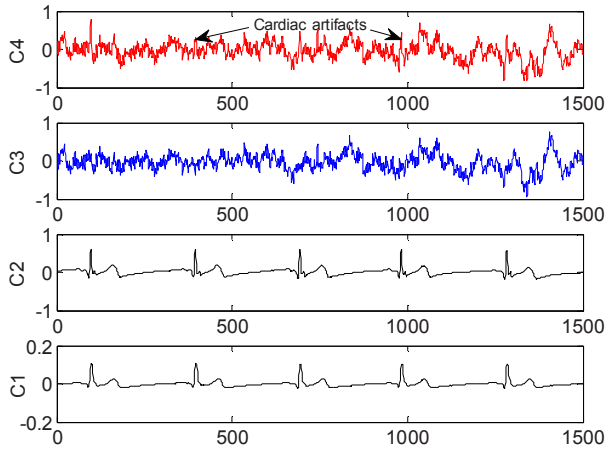

Fig.2. SSP results. C1: ECG reference signal. C2: estimated ECG contribution in MEG signal. C3: denoised MEG. C4: corrupted MEG.

6. Conclusion

This paper investigated the performance of kernel LMS methods KLMS, SSP and NORMA on time series prediction and MEG signal denoising. It showed that KLMS, SSP and NORMA yielded equivalent good performances in terms of minimum NMSE and rate of convergence. The results for NORMA were obtained with a regularization parameter equal to zero, incurring the loss of sparsity of the model. SSP provided substantially sparser solution than KLMS and NORMA. This may attributed to the use of the redundant samples in adjusting the filter parameters. In KLMS, only the centers are involved in the derivation of the solution. Redundant samples are excluded. A significant amount of input samples were admitted in the dictionary and participated in the determination of the filter parameters. Two interesting observations can be drawn from this study: i) the regularization parameter is useless in kernel LMS algorithms. ii) redundant samples have a core role and should be incorporated in the parameter update process. However they induce additional computational cost. The main issue that should be addressed is how to include redundant data without altering the simplicity of the algorithm. Attempts have been made to overcome this problem [17]. The authors suggested to perform a local update of the parameters, i.e. updated only the coefficient of the closest center to the input data, determined by a quantization technique. This subject deserves in-depth study in order to develop a simple and efficient LMS kernel filters.

References

1. T. Ogunfunmi, Adaptive nonlinear system identification: The Volterra and Wiener model approaches, Springer, 2007.

2. S. Haykin, Neural networks and learning machines, Prentice Hall, 2008.

3. I. Constantin, C. Richard and R. Lengelle, Nonlinear regularized Wiener filtering with kerneks: Application in denoising MEG data corrupted by ECG, IEEE Trans. Signal Processing. 54(12): 4796- 4806 (2006).

4. Y. Engel, S. Mannor and R. Meir, The kernel recursive least squares algorithm, IEEE Trans.Signal Processing. 52(8): 2275-2285 (2004).

5. C. Liu and F. Liu, The short-term load forecasting using the kernel recursive least-squares algorithm, 3rd Int. Conf. Biomedical Engineering and Informatics. 2673-2676 (2010).

6. T. J. Dodd, V. Kadirkamanathan and R. F. Harrison, Function estimation in Hilbert spaces using sequential projections, Proc. IFAC Int. Conf. Intelligent control systems and signal processing, University of Algrave, Portugual. 113-118 (2003).

7. S. Phonphitakchai and T.J. Dodd, Stochastic meta descent in online kernel methods, 6th Int. Conf. Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. 2:690-693 (2009).

8. J. Kivinen, A. J. Smola and R. C. Williamson, Online learning with kernels, IEEE Trans. Signal Processing. 52(8): 2165-2176 (2004).

9. S. Smale and Y. Yao, Online learning algorithms, Foundations of Computational Mathematics. 6:145-170 (2006).

10. C. Richard, J. C. M. Bermudez and P. Honeine, Online prediction of time series data with kernels, IEEE Trans. Signal Processing. 57(3) 1058-1066 (2009).

11. W. Liu, J. C. Principe and S. Haykin, Kernel adaptive filtering, Wiley & Sons, Hoboken, New Jersey, 2010.

12. B. Scholkopf and A. Smola, Learning with kernels, MIT PRESS, Cambridge, MA, 2002.

13.Y. Yu, H. Cheng, D. Schuurmans and C. Szepesvari, Characterizing the representer theoerm, Proc. 30th Int. Conf. Machine Learning. 570-578 (2013).

14. J. Platt, A resource allocating network for function interpolation, Neural Computation. 3(2) 213-225 (1991.)

15. S. Chen, Local regularization assisted orthogonal least squares regression,Neurocomputing. 69:559-585 (2006).

16. R. H. Kwong and E. W. Johnston, A variable step size LMS algorithm, IEEE Trans. Signal Processing. 40(7)1633-1642 (1992).

17. B. Chen, S. Zhao, P. Zhu and J. C. Principe, Quantized kernel least mean square algorithm, IEEE Trans. Neural Netw. 23(1) 22-31 (2012).